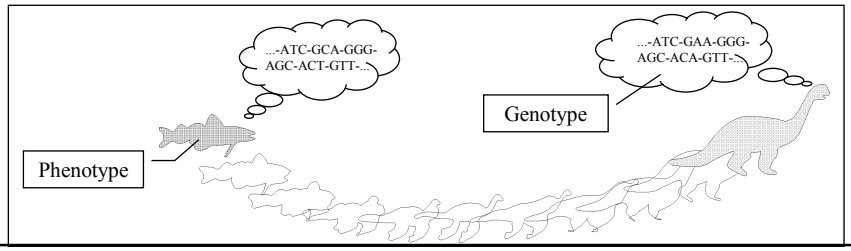
EVOLUTIONARY ALGORITHMS (1)

Basic notions
Genetic algorithms
Evolutionary strategies

Jakub Wróblewski, jakubw@qed.pl

FUNDAMENTAL NOTIONS

Individual – fundamental item of evolution. We usually assume that it lives in a certain **environment**, to which it can be better or worse adjusted. "Goal" of evolution is to create an individual adjusted to environment in a possibly optimal way.
Phenotype – features of an individual, which occur „outside".
Genotype – "construction plan" – complete and unique description of an individual, encoded in its genes.
Population – ensemble of individuals living in a common environment and competing for its resources.



FUNDAMENTAL PRINCIPLES

- Genotype of an individual does not change. However, genotype of its children is usually different. Modifications can be caused by small, random mutations, or by mixing (crossing) with genotype of another parent
- Changes in genotype imply changes in phenotype and thus – also changes in its adjustment to environment. Other changes in phenotype do not affect genetic material of children

FUNDAMENTAL PRINCIPLES

- Changes in genotype are casual. Positive changes are assumed to occur with the same frequency as negative or invariant ones.
- Individuals are evaluated by means of comparison how they fit the environment. These, which are better adjusted, have more chances for children. These with worse fitness usually loose and die.

APPLICATION:
OPTIMIZATION PROBLEMS

Individual (phenotype)	⟷	A solution
Chromosome (genotype)	⟷	Encoded solution
Adjustment to environment	⟷	Function being optimized

Changes (mutation, cross-over) are concerned with genotype, while selection corresponds to phenotype. Essence of evolution is to combine random, unsupervised operations over genotype with strictly supervised influence of environment on phenotype.

HISTORY

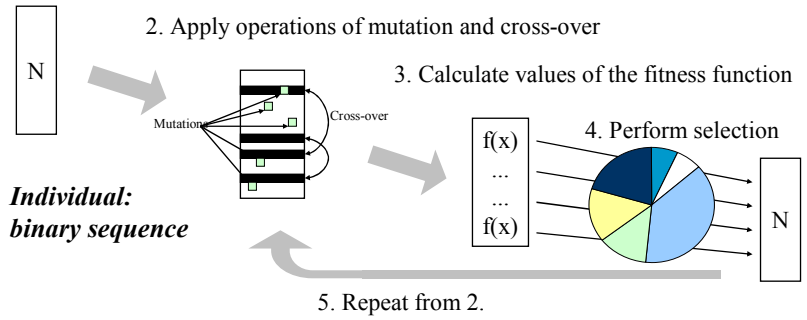
1958, 1964 (Friedberg, Fogel) – **evolutionary programming** over finite automata.
1965 (Bienert, Rechenberg, Schwefel) – **evolutionary strategies**, practical applications.
1975 (Holland) – **genetic algorithms** and their theory.
The 80's – a number of applications of genetic algorithms.
The end of the 80's (Fogel) – currently reigning version of evolutionary programming.

- Z. Michalewicz. *Genetic algorithms + Data structures = Evolutionary programs.*
- D.E. Goldberg. *Genetic algorithms and their applications.*
- J.R. Koza. *Genetic Programming: On the Programming of Computers by Means of the Natural Selection*.

GENETIC ALGORITHM

Goal: Find maximum of function f(x)
Assumption: function is positive

1. Generate N individuals randomly



5. Repeat from 2.

SCHEME OF WORK (1)

The initial step: the problem encoding

Individual – binary sequence of a constant length

In purpose of solving a specified task one has to encode the space of states (i.e. all possible solutions) in a binary language.
If a given task is concerned with finding maximum of a function, then we can use this function to express the degree of adjustment of an individual to environment. Often, however, we have to define such a function by ourselves.

The second step: mutation and cross-over

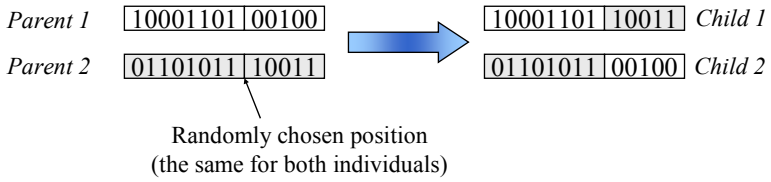
Mutation: Randomly choose an individual and then – one of its bits. Change the value of this bit. Mutation should influence around 0.1% bits in population.

100100110010 ⇒ 100101110010

SCHEME OF WORK (2)

Cross-over (crossing-over)

Join individuals in pairs. For each pair decide (randomly, with probability 0.2 – 0.5), whether they are going to cross. If yes, choose randomly a position (bit) within the chromosome of one of the parents. Then exchange the fragments of chromosomes starting with this position.



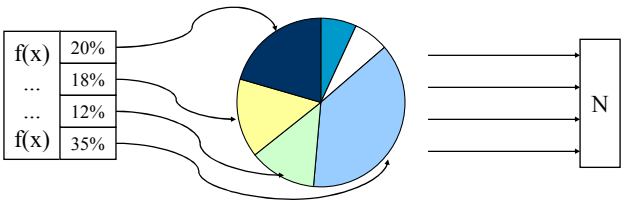
Randomly chosen position
(the same for both individuals)

SCHEME OF WORK (3)

The forth step: selection

Given the values of fitness function for particular individuals, randomly choose N individuals (with repetitions), according to the following algorithm, called „**Roulette Wheel**":

1. Calculate the sum: $f_{sum} = f(x_1)+...+f(x_N)$.
2. Calculate relative fitness: $p(x_i) = f(x_i)/f_{sum}$
3. Given probability distribution $(p(x_1),..., p(x_N))$, choose randomly an individual. Repeat it N times.



THEORY (1)

- **Schemata**:
  - Sequences of the form 100**1*1** (where star means „anything”)
- **The Schemata Theorem**:
  - If a given sequence corresponds to individuals being, averagely, better than the others, then its representatives are going to occur more and more frequently (the exponential growth) during evolution

THEORY (2)

- **Hidden Parallelism**:
  - In a single step we deal with just several individuals. However, each of them corresponds to exponential number of different schemata
- **The Building Blocks Hypothesis**:
  - If the solution of a specified task can be encoded by using strong, short schemata, then the corresponding genetic algorithm is going to find it with a high probability

ADVANTAGES (1)

- **Universal**: In purpose of applying the same implementation to another task, it is usually enough to the change definition of fitness function
- **Robust**: It can cope with situations when the function being optimized is noised, changes in time, has many local maxima

ADVANTAGES (2)

- **Black box style**: One doesn't need to know much about the function being optimized while searching for its maximum
- **Fast**: It is often possible to find the solution after examining surprisingly small number of states
- **Randomized**: Since it performs many operations randomly, one can try to repeat evolution hoping for obtaining better results

DISADVANTAGES

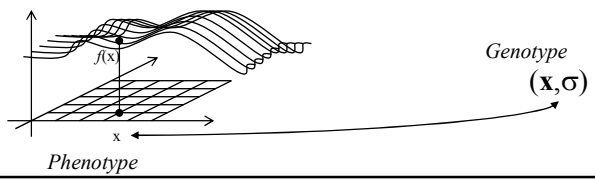
- Since the method is universal, it may perform not as well as more specialized approaches (a solution is to apply hybrid algorithms)
- The success depends on the way of the task encoding, as well as the choice of fitness function and genetic operators. Unfortunately, there is no theory about it (one can just improve her/his experience)
- Because the method is randomized, we cannot be sure whether the obtained solution is optimal indeed (on the other hand, it is better to have an approximately optimal solution than nothing)

EVOLUTIONARY STRATEGIES

Another – besides genetic algorithms – popular evolutionary approach. The main area of applications: optimization of multidimensional real functions.

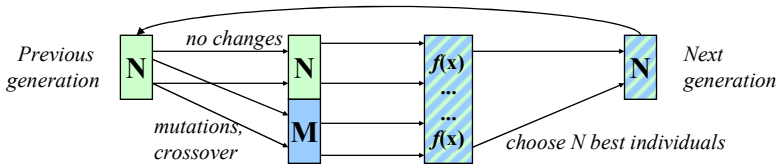
Task: Find maximum of function $f: R^n \rightarrow R$ within the cube $[a_1, b_1] \times ... \times [a_n, b_n]$.

Solution: Let $\mathbf{x}=(x_1, ... x_n)$ denote an exemplary solution (an element of the space of states, i.e. such that $x_i \in [a_i, b_i]$). Attach to each vector $\mathbf{x}$ an additional vector of positive real values $\sigma=(\sigma_1, ... \sigma_n)$. Pair $(\mathbf{x},\sigma)$ is the genetic code of the individual



SCHEME OF WORK

1. Create random population of N individuals, i.e. pairs (**x**,σ).
2. Expand the population by adding M children, created as the result of mutation and crossover.
3. Calculate fitness function for every individual from the expanded population.
4. Select the best N out of N+M individuals from the intermediate population.
5. Repeat from 2 using the new population found (N individuals).



EVOLUTIONARY OPERATORS

Mutation is usually applied to the whole intermediate population. It is done by the following change of parameters (x,σ):

$$\sigma_i = \sigma_i \cdot e^{N(0,\Delta)}$$

$$x_i = x_i + N(0,\sigma_i)$$

where N(0,σ) is a random number drawn from the normal probability distribution (mean 0, standard deviation σ).

Crossover is applied to about 25% of the intermediate population. It is similar to "uniform crossover" operator from the classical genetic algorithm - every *x* value is taken from the random parent.

$(x_1, x_2, x_3, x_4, x_5, \sigma_1, \sigma_2, \sigma_3, \sigma_4, \sigma_5)$

$(y_1, y_2, y_3, y_4, y_5, \tau_1, \tau_2, \tau_3, \tau_4, \tau_5)$

$\rightarrow (x_1, y_2, y_3, x_4, y_5, \sigma_1, \tau_2, \tau_3, \sigma_4, \tau_5)$

Note: value x_i is inherited always together with the corresponding parameter σ_i.